**Market Guide for API Testing and Service Virtualization**

**6 July 2015** ID:G00274577

**Analyst(s):** Thomas E. Murphy, Maritess Sobejana, Joachim Herschmann, Laurie F. Wurster

**VIEW SUMMARY**

Demand for service API testing and virtualization solutions will quickly increase with faster delivery of new business functionality to more complex IT environments. This research highlights important market trends and attributes to help heads of development and testing adopt the right solution.

http://na2.www.gartner.com/imagesrv/reprints/images/bottom.gif;wa3a3395f12ff5357f

**Overview**

**Key Findings**

* The need for API testing and service virtualization is becoming increasingly important as IT organizations develop bimodal capability to support mobile, digital business transformation initiatives and the Internet of Things (IoT).
* The rapid adoption of agile and SOA-based solutions to support business agility demands that development and test organizations change approaches and tools to drive a broader, more accurate view of complex system quality.
* Service virtualization enables earlier, continuous, complete and parallel development and testing of software in a highly complex, heterogeneous and distributed environment. It can also reduce overall test lab costs.
* The market for testing and service virtualization is currently sized around US$200 million and growing at a CAGR of 12%. Growth is steady as businesses open up and share services internally and with partners.

**Recommendations**

Heads of development and testing should:

* Evaluate your current software development and delivery processes, identifying the biggest issues and challenges from process and skills perspectives. This will lead toward an effective solution and guide the required skills to develop or acquire.
* Involve roles from different functions and share ownership across several competency leaders in adopting technologies and tools. This will help identify key obstacles to overcome and provide work-arounds so the project can move forward smoothly.
* Develop a plan for acquiring technologies and tools that will enable much earlier testing in the development life cycle, continuous integration and faster release processes.
* Adopt a long-term tooling strategy that factors in business needs and requirements, existing processes, tool requirements, benefits, costs, expertise and training.

**Market Definition**

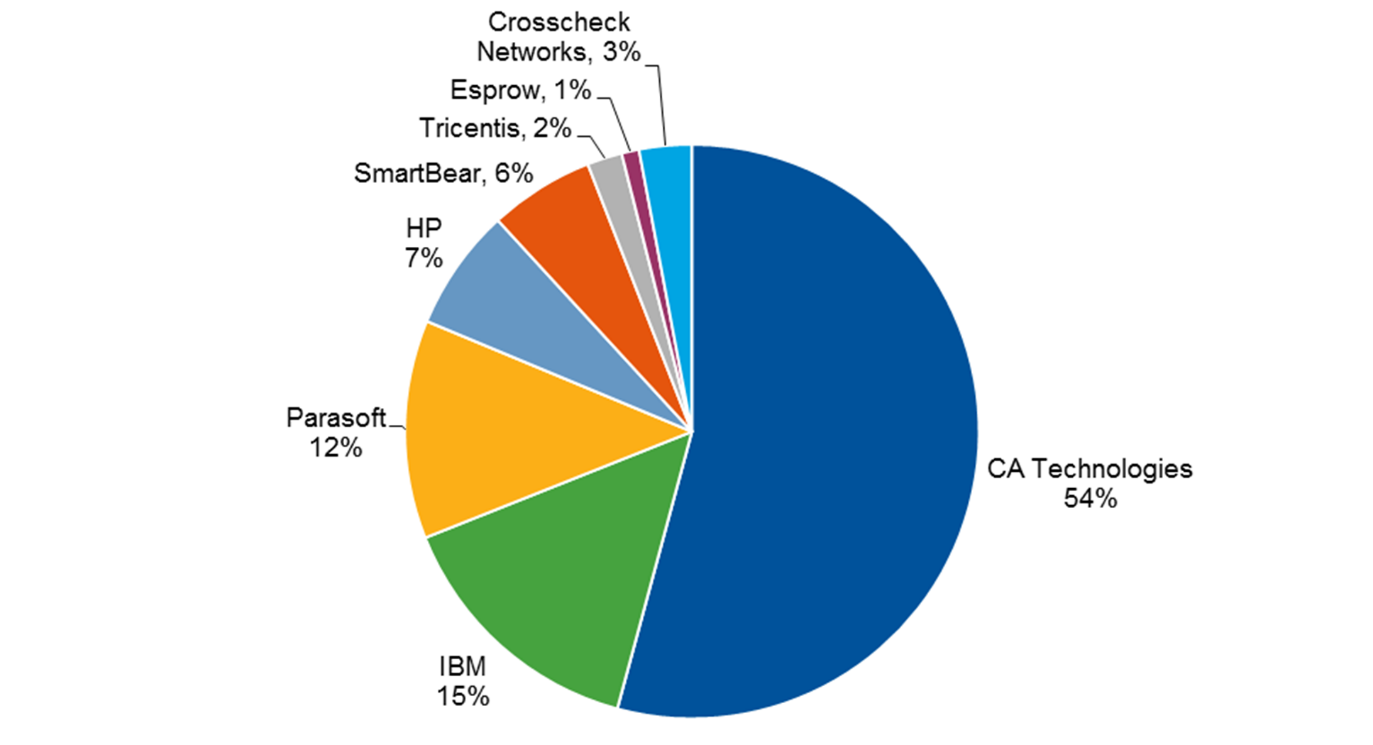
The application programming interface (API) testing and service virtualization market is a subset of the overall software testing tools market. It consists of vendors offering solutions for automating the testing functionality and/or performance for APIs or Web services and solutions, to simulate or "virtualize" interdependent components. Rather than driving the user interface as most automation tools do, these tools test the system-to-system interactions driving the APIs and services. Virtualization is a technique that expands on the concept of [object mocking](http://www.mockobjects.com/) to simulate the behavior of services by adding capabilities such as timing responses, error handling and the ability to simulate unavailable resources. This reduces system costs and allows a deeper set of tests to be run by playing with service parameters.

Service virtualization was first introduced to the market by ITKO (acquired by CA Technologies) in 2007, and early entrants have been the most dominant in the market.

**Market Direction**

Gartner sizes the overall software testing tools market at $1.5 billion, with a 5% growth rate in 2014 (see "Market Share: All Software Markets, Worldwide, 2014" for details). Overall, the API testing and service virtualization submarket has grown steadily as additional vendors have entered (for additional detail, see "Magic Quadrant for Integrated Software Quality Suites"). The initial burst was driven primarily by ITKO (acquired by CA) and Green Hat (acquired by IBM) and, just as it seemed that the market could consolidate, a new set of entrants appeared. The early entrants have seen the largest adoption of their tools. Figure 1 shows current vendor market share.

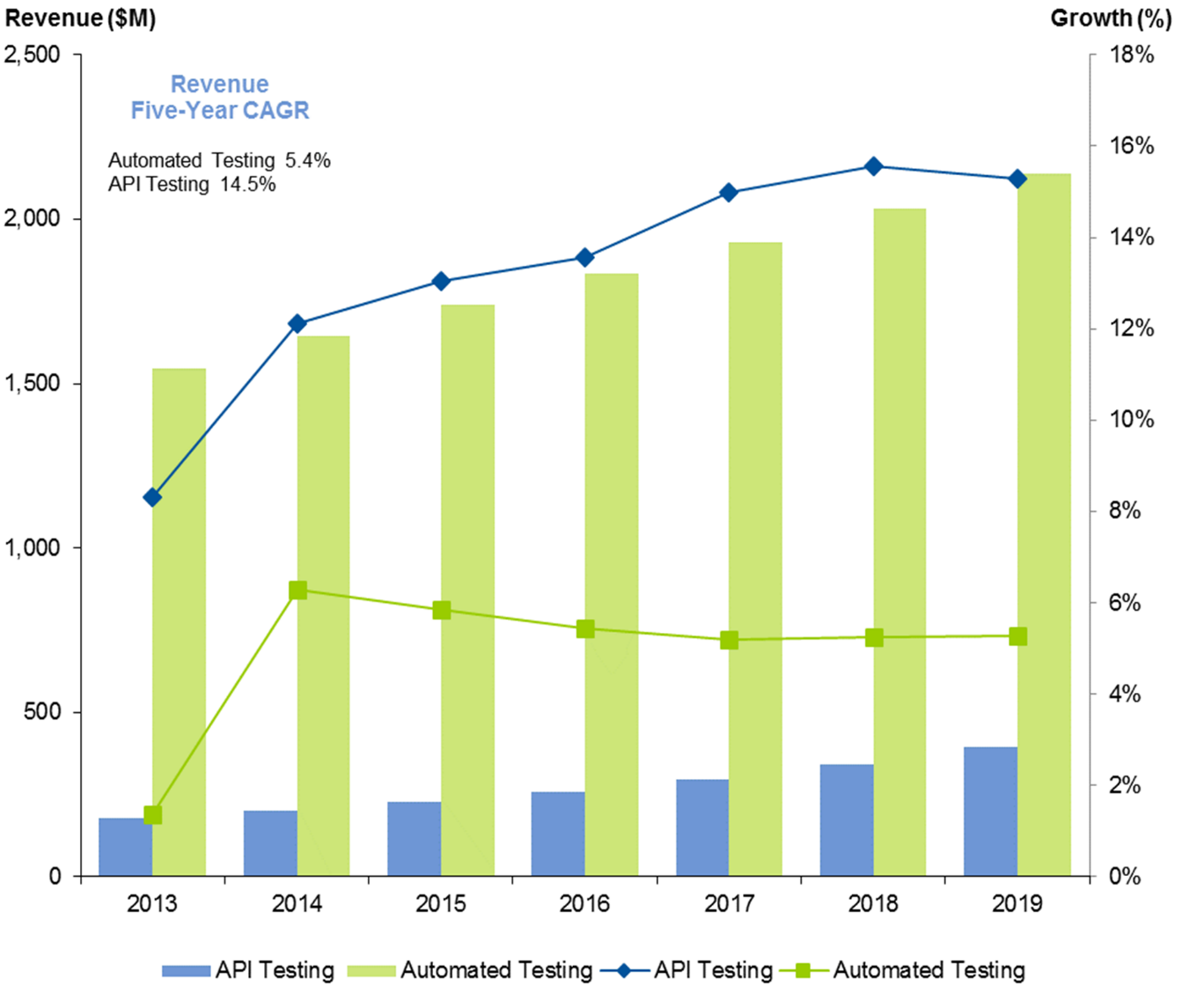
**Figure 1.** Vendor Market Share, 2015



Source: Gartner (July 2015)

We project that there are still opportunities for additional products to enter the market and that the number of variations will increase (for example, database virtualization from Delphix). We estimate the API testing and service virtualization submarket to currently be around $200 million, and forecast a growth rate of 15% by 2019 (see Figure 2).

**Figure 2.** Market Growth — API Testing vs. Overall Software Testing Market, 2013-2019



Enlarge

Source: Gartner (July 2015)

**Market Analysis**

The shift to more service-oriented architectures that heavily rely on dependent services, and the rapid adoption of agile software development practices make it increasingly important to look beyond traditional functionality and performance testing, as well as drive a need for testing earlier and more frequently. By eliminating constraints in today's highly complex composite applications, which are developed using heterogeneous technologies and managed and owned by distributed teams, many software development life cycle activities can happen earlier and become parallel again. Each component owned by a team can be tested individually in the context of its real environment, instead of waiting for issues to surface later during integration, when they are costlier to fix. In testing APIs or services, team members can use a more flexible alternative to developing and maintaining method stubs or mock objects that act as substitutes for dependent resources. Service virtualization further extends this concept by productizing the creation of stubbing frameworks to model the behavior of complex, interdependent services.

When organizations shift toward Web-scale architectures, which are based upon microservices, businesses become more responsive. However, the challenge here is the dependent underlying applications and/or services' availability, which is highly important during integration in parts as well as end-to-end testing of business processes as a whole.

Service virtualization enables development and QA teams to simulate and model the behavior of complex, interdependent unavailable and/or limited services, thus removing constraints in needing to access components, databases, mainframes and so on. Intelligent virtual services can be used all the way from implementation to delivery, and provide significantly more value than simplistic stubs whose use is mainly limited to unit tests in early development. API tests validate the correct behavior of a service interface by generating requests and inspecting the corresponding responses. Simultaneously, virtual services receive these requests, emulate the real-world service's behavior and provide the appropriate responses on any number of supported protocols, in a variety of message formats. Service virtualization shifts the ability to test applications earlier in their development life cycle, enabling integration and release processes to happen faster, with higher quality and less risk. It also enables other nonfunctional testing, such as performance testing against a simulated connection or load testing that simulates multiple connections.

A simple approach of mocking a service can aid a developer with unit testing, but it does not scale to support the entire team. Mocking functions are typically very context-specific, simulating a specific behavioral response to fulfill a certain development need at a certain time (for example, resolving a missing dependency). Developing a realistic simulation to support more test cases is complicated and can quickly become a major development and maintenance effort, diverting time from developing application functionality. In addition, no infrastructure exists to support sharing these stubs across the team. Virtualized services, however, allow team members to emulate environments and make their behavior available to the entire team. They can be deployed throughout the entire production cycle, consistently delivering the same behavior and functionality for anyone who wants to use them at any time. Virtual components that are part of a projectwide test environment eliminate the need for individual developers to write and rewrite their own mocks and stubs, saving time and effort for all team members. From a testing perspective, the behavior of individual classes can be tested with object mocking, whereas the behavior of entire network back-end services can be simulated and tested with service virtualization. As a consequence, mocking is best-suited for unit tests while service virtualization is better for integration and performance tests.

Virtual services can be created and maintained in a number of different ways. One common way is capturing live traffic by listening to the transactions between the various participating systems. Calls and responses between servers, integration layers and other components are intercepted and recorded on the protocol level — for example, HTTP, SOAP, JMS, JDBC, etc. A model of the service behavior can also be created from historical data recorded in a file — for example, a WSDL definition, a day's history of server logs or a spreadsheet of data. User modeling, or manual creation of the conversation between the service components, represents yet another option where the virtualized service is created from scratch.

Established commercial vendors offer both service/API testing and service virtualization capabilities. The natural synergy between automating API tests and service virtualization enables the tools to share protocol definitions and datasets between virtual services and automated service tests. The same test suites can be used to validate both production and virtual components.

On top of the commercial offerings are several open-source tools that can be considered for testing API services and building service virtualization quickly. These tools have limited features but are worth evaluating for fitness of project purpose and requirements.

Several of the most important capabilities and features associated with these tools include:

* Seamless integration between service virtualization solution and the service/API testing tool.
* Broad native support for protocols, technologies and message formats (REST, SOAP, JMS, JSON), security standards (OAuth, WS-Security, Kerberos, SSL, SAML) and industry-specific protocols (FIX, Swift, ISO 20022, ISO 8583).
* Runs on different platforms (Windows, Unix, Linux, Mac OS).
* Integration support with application development life cycle management, IDE and/or testing tools (Eclipse, Visual Studio, HP Quality Center Enterprise, Jira).
* Support for different types of testing such as functional, performance, security, compliance and coverage analysis.
* Comprehensive reporting and dashboard.

**Representative Vendors**

*The vendors listed in this Market Guide do not imply an exhaustive list. This section is intended to provide more understanding of the market and its offerings.*

This list contains two groups of solutions:

1. Commercial offerings that provide both service/API testing as well as service virtualization capabilities — see Table 1.
2. Open-source solutions that provide service/API testing with some limited service virtualization — see Table 2.

Key differentiation in the commercial products includes some of the elements listed in the prior section.

| **Table 1.** API Testing and Service Virtualization — Commercial | | |
| --- | --- | --- |
| **Vendor/Website** | **Products** | **API Testing and Service Virtualization** |
| [CA Technologies](http://www.ca.com/sv) | CA Application Test CA Service Virtualization | ✔ |
| [Crosscheck Networks](http://www.crosschecknet.com/) | SOAPSonar CloudPort | ✔ |
| [Esprow](http://www.esprow.com/) | Nozomi Studio and Nozomi Execution Server Nozomi Studio and Nozomi Simulation Server | ✔ |
| [HP](http://www.hp.com/) | HP Server Test HP Service Virtualization | ✔ |
| [IBM](http://www-03.ibm.com/software/products/en/rtvs/) | IBM Rational Integration Tester IBM Rational Test Virtualization Server | ✔ |
| [Parasoft](http://www.parasoft.com/) | SOAtest and Environment Manager Virtualize and Environment Manager | ✔ |
| [SmartBear](http://smartbear.com/product/ready-api/servicev/overview/) | Ready! API SoapUI NG Pro ServiceV Pro VirtServer | ✔ |
| [Tricentis](http://www.tricentis.com/) | Tosco Testsuite Orchestrated Service Virtualization | ✔ |

Source: Gartner (July 2015)

**Open Source**

Open-source options have tended to focus on lower-level solutions such as object mocking and/or Web-service testing. They do not have the breadth of protocol support or the ability to truly virtualize the service.

| **Table 2.** API Testing and Service Virtualization — Open Source | | |
| --- | --- | --- |
| **Product** | **API Testing** | **Object Mocking** |
| [Apache JMeter](http://jmeter.apache.org/) | ✔ |  |
| [SoapUI](http://www.soapui.org/) | ✔ |  |
| [PowerMock](https://code.google.com/p/powermock/) |  | ✔ |
| [EasyMock](http://easymock.org/) |  | ✔ |
| [REST Assured](https://code.google.com/p/rest-assured/) | ✔ |  |
| [WebInject](http://www.webinject.org/) | ✔ |  |
| [Mockito](http://mockito.org/) |  | ✔ |
| [Jmockit](http://jmockit.org/) |  | ✔ |
| [mocktail](http://mocktail.svashishtha.com/) |  | ✔ |
| [WireMock](http://wiremock.org/) |  | ✔ |
| [Betamax](http://freeside.co/betamax/) |  | ✔ |
| [MockServer](http://www.mock-server.com/) |  | ✔ |

Source: Gartner (July 2015)

**Market Recommendations**

Automating service and API testing and adopting service virtualization technology are not just about tooling; they require development and QA teams to have a detailed understanding of the underlying SOA principles and technologies. The technical nature of the communication protocols, heavy reliance on tools and lack of user-friendly interfaces mean that the traditional approach to QA is inappropriate and, therefore, a transformation of QA practices will be needed to maintain quality of service. QA teams specifically need to acquire new skills on new technologies.

Before adopting service virtualization technology, consider:

* The current state of your software development and delivery processes — that is, identify the biggest challenges or pain points that are costing the organization:
  + Test cycle delays due to unavailability of components connected to applications under test, affecting time to market
  + Access to dependent systems such as mainframe, services, third-party applications, partner software and so on
  + Financially, in terms of costs associated with provisioning test environments
* The cost to implement a service virtualization solution, including:
  + Product licenses — an open-source advantage
  + Hardware cost to host the solution versus cost of hardware to replicate production
  + Learning curve
  + Training — open source relies on community
  + Any associated implementation costs

The above will help develop the business case. Note, however, that service virtualization is just one side of the virtualization equation and forms part of continuous testing process. Virtualization of test data, network and automation of lab environments must not be taken for granted. Service virtualization can work without these, but the ROI will be compromised.

Prior to choosing tools, put in place correct planning and processes. Adopt a long-term tool strategy that factors in existing processes, tool requirements, benefits, costs, expertise and training. Consider carrying out a proof of concept to assess the potential benefits.

Adoption of these technologies and tools will be most effective and successful when the team can participate and ownership is shared across several competency leaders. Organizations need to consider which roles will be involved in various ways, and this includes architects, development leads, QA leads and the operations lead. This will help identify obstacles to overcome and provide work-arounds so the project can move forward smoothly.

We expect that the prevalence of mobile and cloud computing, the emergence of digital business and the IoT will continue to drive adoption of service/API testing and service virtualization solutions. The IoT, in particular, will evolve and create innovation in these products, and vendors will have to broaden their support for IoT protocols as they become adopted in consumer-facing IT solutions.